Examiner's Statement of Reasons for Allowance

1. This action is responsive to Applicant's amendment filed on April 09, 2008.

Examiner's Amendment

2. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Mr. Chad R. Walsh, Registration Number 43,235, on July 18, 2008; the following amendment is authorized by Mr. Walsh for obviating any potential 101 issues and put the claims in condition for allowance.

The application has been amended as follows:

1. It's OK to enter amendment dated 4/8/2008.

2. On the header of the REMARKS dated 4/8/2008, change the 'Appl. No. 10/695,375' to 'Appl. No. 10/735,513' from pages 2 to 9.

3. Clam 13. (Currently Amended) A system for using incremental generation to develop applications, the system comprising:

one or more computers;

a repository for storing a plurality of metadata development objects corresponding to a metamodel, wherein the development objects comprise a plurality of development objects:

a generator comprising:

a search module configured to identify a first main development object, to identify main development objects related to the first main development objects by an aggregation relationship, wherein an aggregation relationship is a whole-part relationship between an aggregate and one or more constituent parts, and to identify main development objects related to the first main development object by an association relationship, wherein an association relationship represents a connection between ~~between~~ two classes;

a comparator module configured to determine if any identified main development objects have changed; and

a generation module configured to re-generate the first main development object if any identified development objects have changed[;],

wherein the main development objects are persisted in separate files and correspond to file borders; and

wherein identifying main development objects related to the first main development objects by the association relationship comprises identifying main development objects related to the first main development objects by the association relationship by following association relationships only across a single file border to a next main development object.

-- The End --

Examiner's Statement of Reason(s) for Allowance

2. Claims 1-2, 4-7, 10-18, 20-21 are allowed.

3. The following is an examiner's statement of reasons for allowance:
The prior arts of record: **Schloegel** et al., teaches a framework that is provided for generating code for the model-based development of a system. According to the framework, the model-based system is modeled using graphical modeling entities. A modular code generator routine is attached directly to at least one of the graphical modeling entities or to a meta-entity or to a collection of entities. The graphical modeling entities are traversed in order to access specified code generator routines. **Preston** et al., teaches software is automatically generated from one or more predefined functions in accordance with an input statement entered in natural language. Semantically meaningful elements are extracted from the input statement and one or more sets of second semantically meaningful elements are extracted from the predefined functions. At least one of a condition, an action and/or a statement is identified in the input statement. **Little** et al., teaches a system and method for computer code generation that can be used to generate code and configuration files from any data source. In accordance with one embodiment of the invention a Generator Framework provides a common set of standards and APIs through which designs may be input. The purpose of the Generator Framework is to unify the code generation techniques implemented in products such as the Builder products from BEA Systems, Inc., by introducing sufficient abstraction levels. **Moore** et al., teaches a computer-implemented method for synchronizing JAVA with UML in a computer system executing

a repository program. The method comprises the steps of determining if a previous revision of JAVA source exist, and if not; creating a new revision of the JAVA source. Next, the new revision is put in a "created" state and the new JAVA source is stored in the new revision. After this, the new revision is put in a "ready" state. A determination is next made if a previous revision of UML representation exist, and if not; a new revision of the UML representation is created. **Schmitter** teaches an object-oriented software system permits live modification of objects deployed in an execution environment. The system comprises an inspector configured to modify the attributes of deployed object, and a means for selecting and launching inspectors configured for modification of corresponding objects in the execution environment. The system is capable of real-time or live updating of objects in an application or execution environment. **Jennings** teaches a user interface --be it graphical (GUI) or telephony (TUI) to an application is defined by stored interface and feature description documents written in XML and JavaScript, so that the user interface and changes thereto can be effected without access to source code. Interface description documents define the appearance and the behavior of the user interface toward the user, while feature description documents define the interaction of the user interface with the interfaced-to application, both in conformity with a user-interface object model. **Jameson** teaches collection makefile generators generate comprehensive makefiles for processing collections of computer files. In operation, the present collection makefile generator dynamically discovers collection content files, classifies them according to content type and required processing actions, and then generates a makefile for performing those actions. **Misheski** et al., teaches framework for use with object-oriented programming systems provides a

software build system that detects modules that make up a software product, examines each module to determine if it is up-to-date, and automatically updates any modules that require processing. The framework includes a software object of a class called "Product" that comprises a software product to be processed and built. An instance of the Product is comprised of multiple software build objects, each of which is called "Object". **Malone** et al., teaches a computer user interface includes a mechanism for graphically representing and displaying user-definable objects of multiple types. The object types that can be represented include data records, not limited to a particular kind of data, and agents. An agent processes information automatically on behalf of the user. Another mechanism allows a user to define objects, for example by using a template. These two mechanisms act together to allow each object to be displayed to the user and acted upon by the user in a uniform way regardless of type. **Coad** et al., teaches an invention relates a method and systems for generating, applying and defining patterns for software development. The software development tool receives an indication of a pattern, generates software code reflecting the pattern, and stores identification information for the pattern in a comment associated with the generated software code. The software development tool receives an indication of the software element, determines whether the software element is capable of playing the role, and when it is determined that the software element is capable of playing the role, designates that the software element plays the role in the pattern. **Bohrer** et al., teaches a method of developing a software system using Object Oriented Technology and frameworks. The problem of allowing an object to acquire and lose ability and function and to modify responsibilities on an object dynamically or, in other

words, to allow an object to acquire and lose the ability to do things dynamically, is addressed. This problem is solved with a framework to be used for developing a software system, e.g. for a business application. The framework comprises a number of classes which are to be processed by a computer system. **Malone** et al., teaches computer user interface includes a mechanism for graphically representing and displaying user-definable objects of multiple types. The object types that can be represented include data records, not limited to a particular kind of data, and agents. An agent processes information automatically on behalf of the user. Another mechanism allows a user to define objects, for example by using a template. These two mechanisms act together to allow each object to be displayed to the user and acted upon by the user in a uniform way regardless of type. New arts made of record: US 20050091185 A1 by **Koutyrine** et al., teaches a method and system for selectively retrieving locally stored runtime objects in an application development environment. According to one embodiment, a generator fetches a requested runtime object from a locally stored set of runtime objects if the locally stored runtime object is valid, and fetches a server stored runtime object if the locally stored runtime object is invalid and the server stored runtime object is valid. US 20040250259 A1 **Lauterbach** et al., teaches a method and system for selectively generatingruntime objects in an application development environment. According to one embodiment, an invalidation manager receives an indication that a development object has been changed, determines which runtime objects from a set of runtime objects are influenced by the changed development object, and invalidates the influenced runtime objects, and a generator receives a request for a runtime object from the set of runtime objects and regenerates the requested runtime object if the

requested runtime object has been invalidated. US 20040250258 A1 **Raghuvir** et al., teaches a method and system for rule based object navigation. According to one embodiment, a rule engine receives an object for rule evaluation, identifies a rule object to evaluate the received object, the rule object including a navigate function based on an object navigation rule, and invokes the navigate function of the rule object. US 20040010775 A1 **Matsa** et al., teaches a method, system and program product for control of regeneration of pooled objects in an object oriented programming environment. Objects in the pool are regenerated according to various schemes that define dependencies that need to be observed in scheduling a regeneration. US 20040205711A1 **Ishmitsu** et al., teaches a system and method relating to creation of object(s) in an object hierarchy structure is provided. An object generator that in response to a command to expand a node/object provides a initiator node that facilitates generation of a new object within the object hierarchy upon expansion of the node. A display component concurrently displays the initiator node with the expanded node. The initiator node used for creating objects placed (e.g., in-line) within an object hierarchy structure. US 20050216884 A1 **Tchochiev** teaches a method of incrementation is provided, which allows the value of a property associated with a task to be changed (incremented) during consecutive executions of the task. A base generator class containing common functionalities needed by software tools is provided. The base generator class also incorporates the concept of incrementation. Generators for performing a specific task that inherits the base generator class are created. The settings of the generator are customizable prior to execution, either through a user interface or programmatically. However, none of

them, taken alone or in combination, teaches the features in a manner as recited in each of the independent claims 1, 13, and 17.

4. Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

5. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 8:00am - 4:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Chih-Ching Chow/

Examiner, Art Unit 2191

7/18/08

/Wei Zhen/
Supervisory Patent Examiner, Art Unit 2191